eip ESCUELA INTERNACIONAL POSGRADOS

Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

Programación Avanzada Python

LECCIÓN 9

PARALELISMO Y CONCURRENCIA

ÍNDICE

Introducción

Objetivos

Paralelismo y concurrencia

Thread en Python

INTRODUCCIÓN

En este capítulo veremos dos conceptos muy interesantes y a la misma vez avanzados como son el paralelismo y la concurrencia. Estos nos permiten poder controlar como se ejecutan nuestros programas permitiéndonos aumentar la eficiencia en cuanto a tiempo de ejecución se refiere.

OBJETIVOS

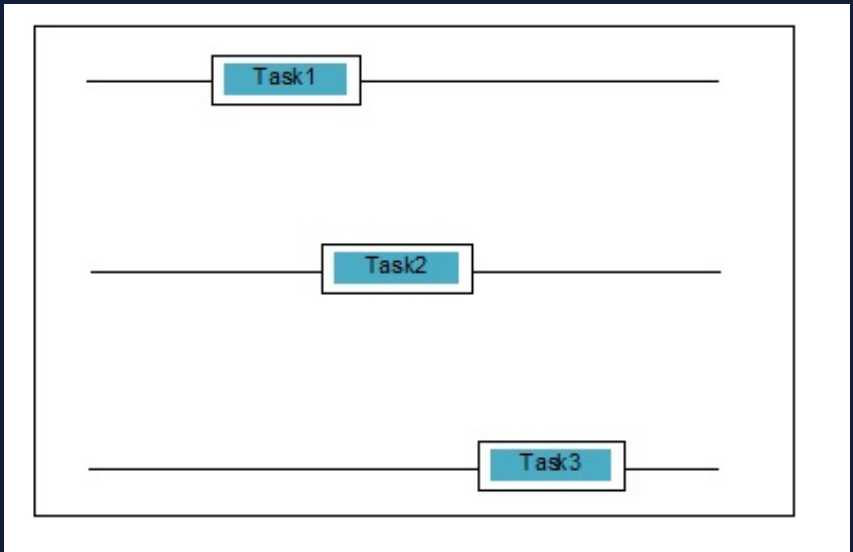
Al finalizar esta lección serás capaz de:

1. Distinguir entre paralelismo y concurrencia
2. Qué paquetes podemos utilizar para el paralelismo y la concurrencia en Python.
3. Cómo trabajar con estos paquetes.
4. Crear un script concurrente

PARALELISMO Y CONCURRENCIA

Concurrencia

La concurrencia es el proceso de múltiples tareas que se ejecutan simultáneamente y ocurre cuando se produce una situación como la superposición de más de una tarea en algunos recursos.



Hay 3 niveles de concurrencia

Concurrencia de bajo nivel:

Realiza acciones atómicas de forma explícita. No es manejada por Python.

Concurrencia de nivel medio:

En lugar de la operación atómica de bajo nivel, utiliza bloqueos explícitamente y es ampliamente utilizada en Python y otros lenguajes. La aplicación puede construirse mediante bloqueos explícitos.

Concurrencia de alto nivel:

Aquí no se usan operaciones atómicas ni se usan bloqueos explícitos. Utiliza un tipo de característica y de modelos concurrentes para realizar la concurrencia en Python.

Propiedades

Correctness:

El programa debe ejecutarse de forma adecuada y correcta, lo que significa que debe de haber una salida en la etapa final.

Seguridad:

Debemos de asegurarnos de que todos los procesos podrán ser ejecutados correctamente.

Live:

El programa debe estar vivo, lo que significa que tiene la capacidad de pasar de un estado a otro.

Propiedades

Actores:

Los procesos y los hilos son actores de los programas concurrentes. Ayudan a hacer una progresión de un estado a otro completando las múltiples tareas simultáneamente.

Recursos:

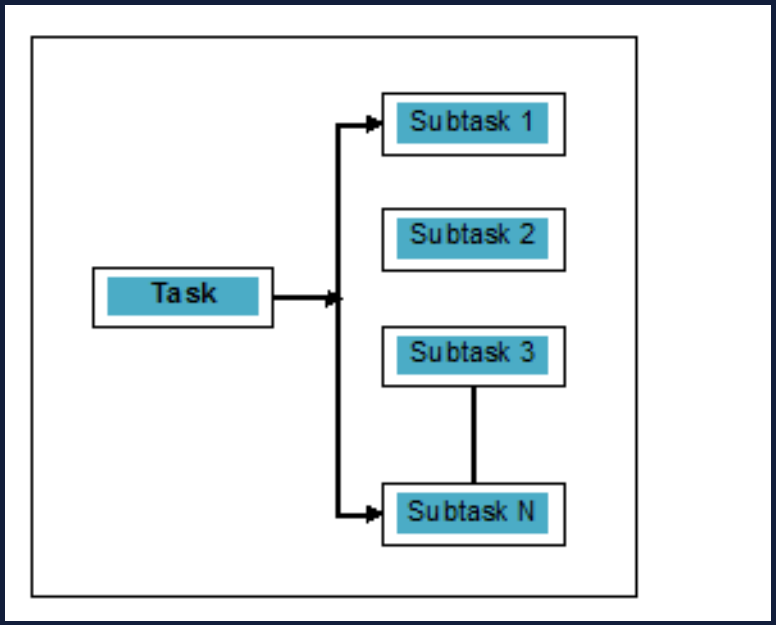
Para realizar las tareas, los actores procesos e hilos utilizan recursos como la memoria, CPU y otras aplicaciones

Conjunto de reglas:

Hay un conjunto de reglas que preceden al programa de forma concurrente como la asignación de memoria, la asignación de recursos, la modificación del estado, etc. Más adelante hablaremos de todo ello en detalle.

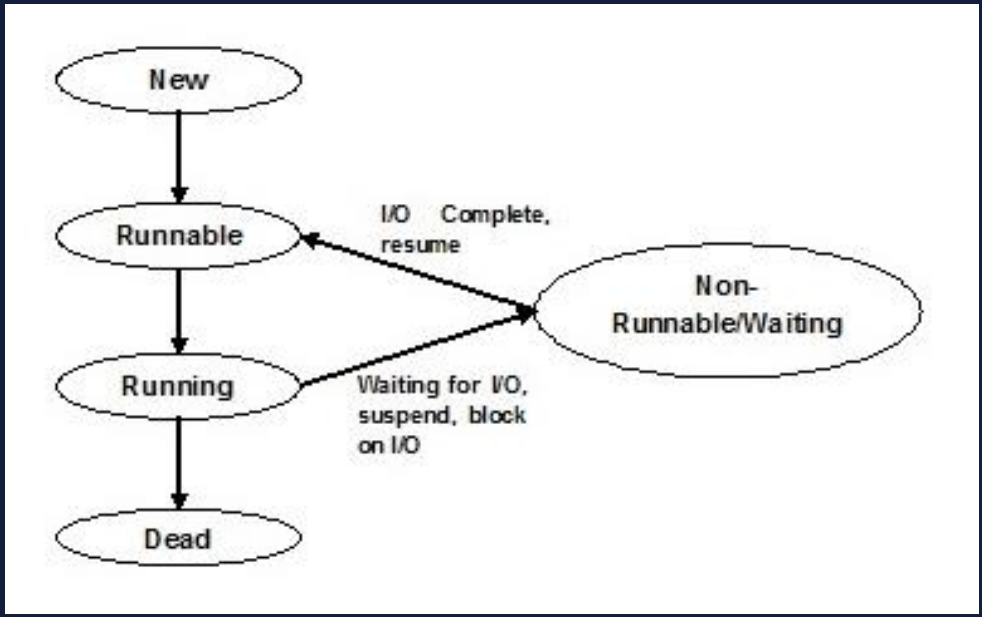
Paralelismo

El paralelismo o la ejecución de tareas al mismo tiempo ocurre como la división de una tarea en algunas subtareas y su ejecución en paralelo.



Threads

Los hilos (threads) son la unidad más pequeña de ejecución, es un solo flujo de control, que se ejecuta dentro del programa.



Estados de los hilos:

Nuevo estado

Comienzo del ciclo, sólo se crea la instancia para un nuevo hilo y no se asigna ningún recurso.

Estado ejecutable

La hebra esta lista para ser ejecutada. Está iniciada y esperando a ejecutarse.

Estado de ejecución

La hebra se está ejecutando, el planificador coge el control y el hilo puede ir a la etapa de espera y al estado muerto.

Estado de espera:

Si ocurre cualquier interrupción desde cualquier otro recurso o proceso, el hilo actualmente en ejecución pasará al estado de espera para completar la interrupción. Después de completar la interrupción pasa al estado Runnable y cuando el planificador lo selecciona pasa al estado Running.

Estado muerto:

Cuando un hilo completa su tarea pasa al estado muerto.

Thread en Python

Módulo _thread

Usando el paquete _thread se ejecuta el hilo como una función.

Con la función de start_new_thread() un nuevo hilo es generado. Esta función tiene dos parámetros:

Función - especificando la función dada como el hilo.

Args[, kwargs] - argumentos de la función especificados aquí en forma de tupla. Si tiene argumentos de palabra clave, puede dar como un diccionario en kwargs.

Sintaxis:

_thread.start_new_thread (function.args [, kwargs]

```
import _thread

def print_num(thread,num):
  for i in range(1,num+1):
    print("Thread", thread,":",i)

try:
  _thread.start_new_thread( print_num, (1,3))
  _thread.start_new_thread( print_num, (2,5))
except:
  print("Error")
```

Resultado:

```
Thread 2 : 1
Thread 2 : 2
Thread 2 : 3
Thread 2 : 4
Thread 2 : 5
Thread 1 : 1
Thread 1 : 2
Thread 1 : 3
```

Módulo threading

Este paquete tiene mayores capacidades que el módulo _thread. Este paquete trata los hilos como un objeto o instancia. Threading tiene las funciones que se encuentran en _thread y también agrega algunas otras funciones como las siguientes:

- activeCount () - número activo de hilos devueltos
- currentThread ()- devuelve el número de hilo.
- enumerate ()- devuelve la lista de todos los hilos que están actualmente activos.

Módulo threading

Para implementar hilos utilizando este módulo, debemos de utilizar la clase thread con algunos métodos:

- run () - punto de entrada de un hilo
- start ()- inicia un hilo llamando a run ()
- join([time]) - espera la salida del hilo
- isAlive()- comprueba si el hilo está vivo o sigue ejecutándose
- getName()- devuelve el nombre del hilo
- setName ()- establece el nombre del hilo

Sincronización de hilos

Si hay uno o más hilos, entonces todos los hilos deben estar sincronizados correctamente en el caso de compartir datos. De lo contrario, el resultado final será incorrecto o existe la posibilidad de bloquear los procesos. La sincronización es el proceso por el cual dos o más hilos acceden a los recursos al mismo tiempo sin interferir entre sí.

Los principales problemas que surgen con la sincronización son

- **Bloqueo**:

El caso de que ningún hilo pueda acceder a ningún recurso.

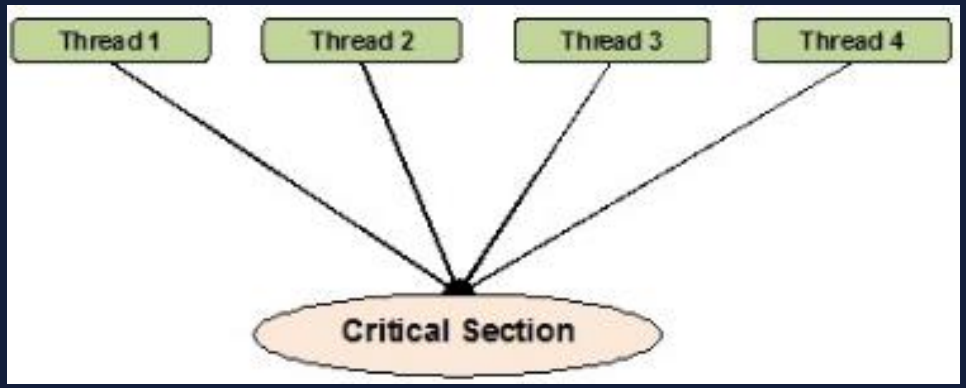
- Condición de carrera:

El caso de que dos o más hilos puedan acceder a los datos compartidos y traten de cambiar su valor al mismo tiempo.

Para hacerlo sin problemas, se utiliza un concepto llamado sesión crítica.

Sección crítica

Es una parte del programa que contiene la variable compartida o el acceso al recurso. Cuando un hilo entra en la sesión crítica, ésta es bloqueada por el hilo y sólo después de completar su ejecución, cualquier otro hilo puede entrar en ella. Así que en la sesión crítica sólo hay un hilo a la vez.



CONCLUSIONES

1. La **concurrencia** es el proceso de múltiples tareas que se ejecutan simultáneamente y ocurre cuando se produce una situación como la superposición de más de una tarea en algunos recursos.

2. El **paralelismo** o la ejecución de tareas al mismo tiempo ocurre como la división de una tarea en algunas subtareas y su ejecución en paralelo.

3. Los **hilos (threads)** son la unidad más pequeña de ejecución.

MUCHAS GRACIAS POR SU ATENCIÓN

eip ESCUELA INTERNACIONAL POSGRADOS

rsanchezi@grupomainjobs.com

Rubén Sánchez Iruela
linkedin.com/in/ruben-sanchez-iruela-8156799a

twitter.com/eiposgrados

facebook.com/eiposgrados

instagram.com/eiposgrados